1.9 Vocabulary

Throughout the book, we try to define each term the first time we use it. At the end of each chapter, we include the new terms and their definitions in order of appearance. If you spend some time learning this vocabulary, you will have an easier time reading the following chapters.

problem solving: The process of formulating a problem, finding a solution, and expressing the solution.

program: A sequence of instructions that specifies how to perform tasks on a computer.

programming: The application of problem solving to creating executable computer programs.

computer science: The scientific and practical approach to computation and its applications.

algorithm: A procedure or formula for solving a problem, with or without a computer.

bug: An error in a program.

debugging: The process of finding and removing errors.

high-level language: A programming language that is designed to be easy for humans to read and write.

low-level language: A programming language that is designed to be easy for a computer to run. Also called "machine language" or "assembly language".

portable: The ability of a program to run on more than one kind of computer.

interpret: To run a program in a high-level language by translating it one line at a time and immediately executing the corresponding instructions.

compile: To translate a program in a high-level language into a low-level language, all at once, in preparation for later execution.

source code: A program in a high-level language, before being compiled.

object code: The output of the compiler, after translating the program.

executable: Another name for object code that is ready to run on specific hardware.

byte code: A special kind of object code used for Java programs. Byte code is similar to a low-level language, but it is portable like a high-level language.

statement: Part of a program that specifies one step of an algorithm.

print statement: A statement that causes output to be displayed on the screen.

method: A named sequence of statements.

class: For now, a collection of related methods. (We will see later that there is more to it.)

comment: A part of a program that contains information about the program but has no effect when the program runs.

string: A sequence of characters; the primary data type for text.

2.11 Vocabulary

variable: A named storage location for values. All variables have a type, which is declared when the variable is created.

value: A number, string, or other data that can be stored in a variable. Every value belongs to a type (for example, `int` or `String`).

declaration: A statement that creates a new variable and specifies its type.

type: Mathematically speaking, a set of values. The type of a variable determines which values it can have.

syntax: The structure of a program; the arrangement of the words and symbols it contains.

keyword: A reserved word used by the compiler to analyze programs. You cannot use keywords (like `public`, `class`, and `void`) as variable names.

assignment: A statement that gives a value to a variable.

initialize: To assign a variable for the first time.

state: The variables in a program and their current values.

state diagram: A graphical representation of the state of a program at a point in time.

operator: A symbol that represents a computation like addition, multiplication, or string concatenation.

operand: One of the values on which an operator operates. Most operators in Java require two operands.

expression: A combination of variables, operators, and values that represents a single value. Expressions also have types, as determined by their operators and operands.

floating-point: A data type that represents numbers with an integer part and a fractional part. In Java, the default floating-point type is `double`.

rounding error: The difference between the number we want to represent and the nearest floating-point number.

concatenate: To join two values, often strings, end-to-end.

order of operations: The rules that determine in what order operations are evaluated.

composition: The ability to combine simple expressions and statements into compound expressions and statements.

compile-time error: An error in the source code that makes it impossible to compile. Also called a "syntax error".

parse: To analyze the structure of a program; what the compiler does first.

run-time error: An error in a program that makes it impossible to run to completion. Also called an "exception".

logic error: An error in a program that makes it do something other than what the programmer intended.

3.11 Vocabulary

package: A group of classes that are related to each other.

address: The location of a value in computer memory, often represented as a hexadecimal integer.

library: A collection of packages and classes that are available for use in other programs.

import statement: A statement that allows programs to use classes defined in other packages.

token: A basic element of a program, such as a word, space, symbol, or number.

literal: A value that appears in source code. For example, `"Hello"` is a string literal and `74` is an integer literal.

magic number: A number that appears without explanation as part of an expression. It should generally be replaced with a constant.

constant: A variable, declared `final`, whose value cannot be changed.

format string: A string passed to `printf` to specify the format of the output.

format specifier: A special code that begins with a percent sign and specifies the data type and format of the corresponding value.

type cast: An operation that explicitly converts one data type into another. In Java it appears as a type name in parentheses, like `(int)`.

modulus: An operator that yields the remainder when one integer is divided by another. In Java, it is denoted with a percent sign; for example, `5 % 2` is `1`.

4.10 Vocabulary

argument: A value that you provide when you invoke a method. This value must have the same type as the corresponding parameter.

invoke: To cause a method to execute. Also known as "calling" a method.

parameter: A piece of information that a method requires before it can run. Parameters are variables: they contain values and have types.

flow of execution: The order in which Java executes methods and statements. It may not necessarily be from top to bottom, left to right.

parameter passing: The process of assigning an argument value to a parameter variable.

local variable: A variable declared inside a method. Local variables cannot be accessed from outside their method.

stack diagram: A graphical representation of the variables belonging to each method. The method calls are "stacked" from top to bottom, in the flow of execution.

frame: In a stack diagram, a representation of the variables and parameters for a method, along with their current values.

signature: The first line of a method that defines its name, return type, and parameters.

Javadoc: A tool that reads Java source code and generates documentation in HTML format.

documentation: Comments that describe the technical operation of a class or method.

5.11 Vocabulary

boolean: A data type with only two values, `true` and `false`.

relational operator: An operator that compares two values and produces a `boolean` indicating the relationship between them.

logical operator: An operator that combines boolean values and produces a boolean value.

short circuit: A way of evaluating logical operators that only evaluates the second operand if necessary.

De Morgan's laws: Mathematical rules that show how to negate a logical expression.

conditional statement: A statement that uses a condition to determine which statements to execute.

branch: One of the alternative sets of statements inside a conditional statement.

chaining: A way of joining several conditional statements in sequence.

nesting: Putting a conditional statement inside one or both branches of another conditional statement.

flag: A variable (usually `boolean`) that represents a condition or status.

recursion: The process of invoking (and restarting) the same method that is currently executing.

recursive: A method that invokes itself, usually with different arguments.

base case: A condition that causes a recursive method *not* to make another recursive call.

binary: A system that uses only zeros and ones to represent numbers. Also known as "base 2".

6.10 Vocabulary

void method: A method that does not return a value.

value method: A method that returns a value.

return type: The type of value a method returns.

return value: The value provided as the result of a method invocation.

temporary variable: A short-lived variable, often used for debugging.

dead code: Part of a program that can never be executed, often because it appears after a `return` statement.

incremental development: A process for creating programs by writing a few lines at a time, compiling, and testing.

stub: A placeholder for an incomplete method so that the class will compile.

scaffolding: Code that is used during program development but is not part of the final version.

functional decomposition: A process for breaking down a complex computation into simple methods, then composing the methods to perform the computation.

overload: To define more than one method with the same name but different parameters.

tag: A label that begins with an at sign (`@`) and is used by Javadoc to organize documentation into sections.

Turing complete: A programming language that can implement any theoretically possible algorithm.

factorial: The product of all the integers up to and including a given integer.

leap of faith: A way to read recursive programs by assuming that the recursive call works, rather than following the flow of execution.

7.8 Vocabulary

iteration: Executing a sequence of statements repeatedly.

loop: A statement that executes a sequence of statements repeatedly.

loop body: The statements inside the loop.

infinite loop: A loop whose condition is always true.

program development: A process for writing programs. So far we have seen "incremental development" and "encapsulation and generalization".

encapsulate: To wrap a sequence of statements in a method.

generalize: To replace something unnecessarily specific (like a constant value) with something appropriately general (like a variable or parameter).

loop variable: A variable that is initialized, tested, and updated in order to control a loop.

increment: Increase the value of a variable.

decrement: Decrease the value of a variable.

pretest loop: A loop that tests the condition before each iteration.

posttest loop: A loop that tests the condition after each iteration.

11.10 Vocabulary

class: Previously, we defined a class as a collection of related methods. Now you know that a class is also a template for a new type of object.

instance: A member of a class. Every object is an instance of some class.

instantiate: Create a new instance of a class in the computer's memory.

data encapsulation: A technique for bundling multiple named variables into a single object.

instance variable: An attribute of an object; a non-static variable defined at the class level.

information hiding: The practice of making instance variables `private` to limit dependencies between classes.

constructor: A special method that initializes the instance variables of a newly-constructed object.

shadowing: Defining a local variable or parameter with the same name and type as an instance variable.

client: A class that uses objects defined in another class.

getter: A method that returns the value of an instance variable.

setter: A method that assigns a value to an instance variable.

override: Replacing a default implementation of a method, such as `toString`.

instance method: A non-static method that has access to `this` and the instance variables.

identical: Two values that are the same; in the case of objects, two variables that refer to the same object.

equivalent: Two objects that are "equal" but not necessarily identical, as defined by the `equals` method.

pure method: A static method that depends only on its parameters and no other data.

modifier method: A method that changes the state (instance variables) of an object.

A.8 Vocabulary

IDE: An "integrated development environment" that includes tools for editing, compiling, and debugging programs.

JDK: The "Java Development Kit" that contains the compiler, Javadoc, and other tools.

JVM: The "Java Virtual Machine" that interprets the compiled byte code.

text editor: A program that edits plain text files, the format used by most programming languages.

JAR: A "Java Archive", which is essentially a ZIP file containing classes and other resources.

command-line interface: A means of interacting with the computer by issuing commands in the form of successive lines of text.

redirection operator: A command-line feature that substitutes `System.in` and/or `System.out` with a plain text file.